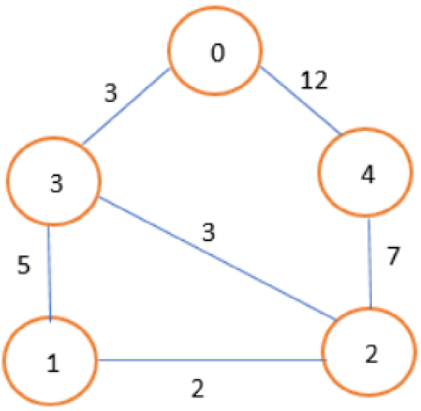
Started on	Tuesday, 15 April 2025, 1:46 PM
State	Finished
Completed on	Tuesday, 15 April 2025, 2:17 PM
Time taken	31 mins 34 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Given a weighted, undirected and connected graph of **V** vertices and **E** edges. The task is to find the sum of weights of the edges of the Minimum Spanning Tree. Write the main function to generate the MST.



Answer: (penalty regime: 0 %)

Reset answer

```
/*
#include<bits/stdc++.h>
using namespace std;
# define INF 0x3f3f3f3f

// iPair ==> Integer Pair
typedef pair<int, int> iPair;

// This class represents a directed graph using
// adjacency list representation
class Graph
{
    int V; // No. of vertices

    // In a weighted graph, we need to store vertex
    // and weight pair for every edge
    list< pair<int, int> > *adj;

public:
    Graph(int V); // Constructor

    // function to add an edge to graph
```

	Input	Expected	Got	
✓	3 3 0 1 5 0 2 1 1 2 3	Prim's MST edges are: 2 - 1 0 - 2 MST cost = 4	Prim's MST edges are: 2 - 1 0 - 2 MST cost = 4	✓

	Input	Expected	Got	
✓	9 14 0 1 4 0 7 8 1 2 8 1 7 11 2 3 7 2 8 2 2 5 4 3 4 9 3 5 14 4 5 10 5 6 2 6 7 1 6 8 6 7 8 7	Prim's MST edges are: 0 - 1 1 - 2 2 - 3 3 - 4 2 - 5 5 - 6 6 - 7 2 - 8 MST cost = 37	Prim's MST edges are: 0 - 1 1 - 2 2 - 3 3 - 4 2 - 5 5 - 6 6 - 7 2 - 8 MST cost = 37	✓
✓	5 5 0 1 3 0 3 3 1 4 4 2 4 1 2 3 2	Prim's MST edges are: 0 - 1 3 - 2 0 - 3 2 - 4 MST cost = 9	Prim's MST edges are: 0 - 1 3 - 2 0 - 3 2 - 4 MST cost = 9	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

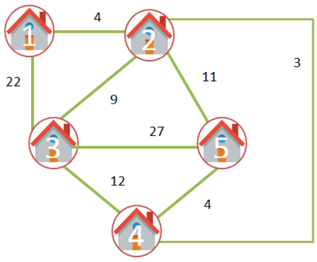
Question **2**

Correct

Mark 20.00 out of 20.00

Given a **houses of a city** consisting of **N** 2D coordinates **{x, y}** where each coordinate represents the location of each house, the task is to find the minimum cost to connect all the houses of the city.

Examples:



Output: 20

Write a CPP function to add edge and weight of the above graph to find MST.

Answer: (penalty regime: 0 %)

Reset answer

```
/*
class Graph {
    vector<vector<int> > edgelist;
    int V;

public:
    Graph(int V) { this->V = V; }
    */
void addEdge(int x, int y, int w)
    {
        edgelist.push_back({ w, x, y });
    }

```

	Input	Expected	Got	
✓	4 5 0 1 10 1 3 15 2 3 4 2 0 6 0 3 5	2 3 4 0 3 5 0 1 10 Minimum Cost Spanning Tree: 19	2 3 4 0 3 5 0 1 10 Minimum Cost Spanning Tree: 19	✓
✓	3 3 1 2 5 1 3 6 3 2 1	3 2 1 1 2 5 Minimum Cost Spanning Tree: 6	3 2 1 1 2 5 Minimum Cost Spanning Tree: 6	✓

	Input	Expected	Got	
✓	5 8 1 2 4 1 3 22 2 5 11 2 3 9 3 5 27 4 5 4 3 4 12 2 4 3	2 4 3 1 2 4 4 5 4 2 3 9 Minimum Cost Spanning Tree: 20	2 4 3 1 2 4 4 5 4 2 3 9 Minimum Cost Spanning Tree: 20	✓

Passed all tests! ✓

Correct

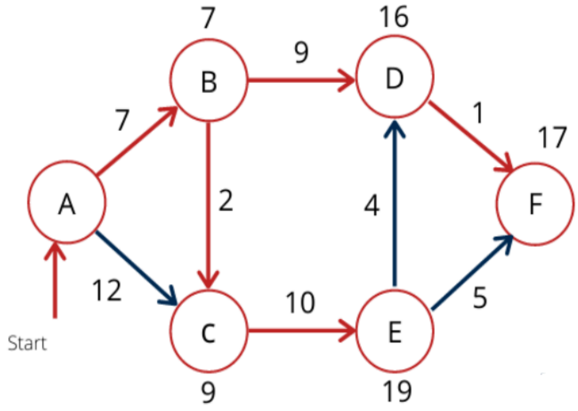
Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a CPP code to **print shortest distances** in the program to find Dijkstra's shortest path from A to all other vertices.



Note: Source is always 0. Give your input as numbers. For example: A B 7 is given as 0 1 7

Answer: (penalty regime: 0 %)

Reset answer

```
/*class Graph
{
    int V; // No. of vertices


public:
    Graph(int V); // Constructor

    // function to add an edge to graph
    void addEdge(int u, int v, int w);
    list< pair<int, int> > *adj;
    // prints shortest path from s
    void shortestPath(int s);
};
Graph::Graph(int V)
{
    this->V = V;
    adj = new list< pair<int, int> >[V];
}

void Graph::addEdge(int u, int v, int w)
{
```

	Input	Expected	Got	
✓	6 8 0 1 7 0 2 12 1 3 9 1 2 2 2 4 10 4 3 4 3 5 1 4 5 5	Vertex Distance from Source A 0 B 7 C 9 D 16 E 19 F 17	Vertex Distance from Source A 0 B 7 C 9 D 16 E 19 F 17	✓

	Input	Expected	Got	
✓	6 9 0 1 7 0 2 9 1 3 15 2 3 11 3 4 6 4 5 9 5 2 2 5 0 14 1 2 10	Vertex Distance from Source A 0 B 7 C 9 D 20 E 20 F 11	Vertex Distance from Source A 0 B 7 C 9 D 20 E 20 F 11	✓
✓	6 9 0 1 4 0 2 5 1 2 11 1 3 9 1 4 7 2 4 3 3 4 13 3 5 2 4 5 6	Vertex Distance from Source A 0 B 4 C 5 D 13 E 8 F 14	Vertex Distance from Source A 0 B 4 C 5 D 13 E 8 F 14	✓
✓	9 14 0 1 4 0 7 8 1 2 8 1 7 11 2 3 7 2 8 2 2 5 4 3 4 9 3 5 14 4 5 10 5 6 2	Vertex Distance from Source A 0 B 4 C 12 D 19 E 26 F 16 G 18 H 8 I 14	Vertex Distance from Source A 0 B 4 C 12 D 19 E 26 F 16 G 18 H 8 I 14	✓

Passed all tests! ✓

Correct

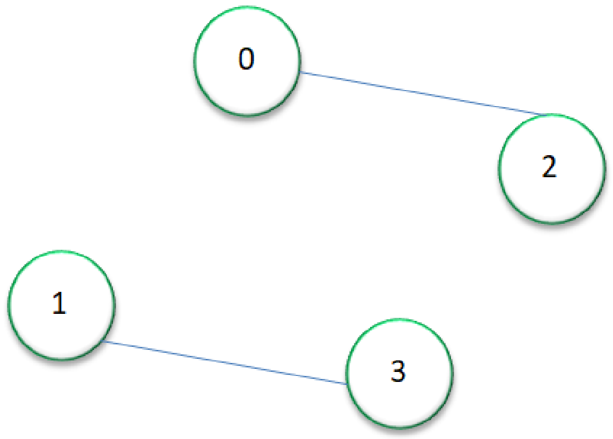
Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Writ a CPP function to find the bipartite of a graph which is given by user.



Answer: (penalty regime: 0 %)

Reset answer

```
/*
#include <bits/stdc++.h>
using namespace std;
class Graph{
    int numVertices;
    list<int> *adjLists;

  public:
    Graph(int V);
    void addEdge(int src, int dest);
};
// Add edge
void addEdge(vector<int> adj[], int s, int d) {
  adj[s].push_back(d);
  adj[d].push_back(s);
}

// Print the graph
void printGraph(vector<int> adj[], int V) {
  for (int d = 0; d < V; ++d) {
    cout << "\n Vertex "
       << d << ":";
```

	Input	Expected	Got	
✓	4 4 1 3 0 2 1 3 0 2	Yes bipartite graph	Yes bipartite graph	✓
✓	9 9 0 1 1 2 1 7 2 3 3 5 4 6 4 8 7 8 1 3	No not a bipartite graph	No not a bipartite graph	✓

	Input	Expected	Got	
✓	9 8 0 1 1 2 1 7 2 3 3 5 4 6 4 8 7 8 9 8	Yes bipartite graph	Yes bipartite graph	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a CPP program to override the print() function in the base class with the print() function in the child class using the concept of virtual functions.

For example:

Test	Input	Result
1	VirtualOne	VirtualOne

Answer: (penalty regime: 0 %)

Reset answer

```
#include<iostream>
#include<string>
using namespace std;
class base
{
    public:
    string a;

    virtual void disp()
    {
        cin>>a;
        cout<<a<<endl;
    }
};
class derive:public base
{
    public:
    void disp()
    {
        cout<<a<<endl;
    }
};
```

	Test	Input	Expected	Got	
✓	1	VirtualOne	VirtualOne	VirtualOne	✓
✓	2	VirtualTwo	VirtualTwo	VirtualTwo	✓
✓	3	VirtualThree	VirtualThree	VirtualThree	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.